**IN THE CLAIMS:**

1. (Currently amended) ~~A computer implemented method for processing encryption requests, said method~~ The method of claim 31, further comprising:

sending~~, in a computer system having a plurality of processors that share a common memory wherein at least two of the processors are dislike,~~ an encryption request from a first processor in the at least one first processor to [[a]] the second processor;

receiving, at the second processor, the encryption request;

reading data from the common memory into [[a]] the local memory ~~corresponding to~~ associated with the second processor, wherein the reading is performed by the second processor ~~and wherein the second processor's local memory is not shared with the first processor~~;

executing at the second processor, an encryption process corresponding to the request, the encryption process being adapted to transform the data; and

writing the transformed data from the second processor to the common memory.

2. (Original) The method as described in claim 1 further comprising:

reading, at the second processor, one or more special nonvolatile registers, the special registers including one or more encryption keys; and

using one or more of the encryption keys in the encryption process.

3. (Original) The method as described in claim 1, wherein the sending further comprises writing the request to a mailbox that corresponds to the second processor and the receiving further comprises checking the second processor's mailbox from the second processor.

4. (Original) The method as described in claim 1 further comprising:

identifying an input data area in the common memory from which the data is read and an output buffer area to which the transformed data is written.

5. (Currently amended) The method as described in claim 1, wherein configuring the second processor further ~~comprising~~ comprises:

~~initializing the second processor prior to receiving the request, the initializing further~~

~~including:~~

reading, from the common memory, initialization software code to be executed on the second processor; and

authenticating the initialization software code.

6. (Original) The method as described in claim 5 wherein the authenticating is performed by a routine stored in a nonvolatile memory and wherein the executing of the encryption process is only performed if the initialization software code is successfully authenticated.

7. (Original) The method as described in claim 6 further comprising:

reading, at the second processor, one or more special nonvolatile registers, the special nonvolatile registers including one or more encryption keys, after the initialization software code is successfully authenticated; and

restricting access to the special nonvolatile registers from outside of the second processor.

8. (Original) The method as described in claim 1 wherein the reading and writing steps are performed using DMA operations.

9. (Currently amended) The method as described in claim 1 further comprising:

identifying the encryption process and an encryption algorithm from a plurality of encryption processes and encryption algorithms based upon the encryption request; and

loading encryption software code corresponding to the identified encryption process and the encryption algorithm, the loading <u>being</u> performed by reading the encryption software code from the common memory to the second processor's local memory.

10. (Original) The method as described in claim 1 wherein the encryption process is selected from the group consisting of a decryption function, an encryption function, and an authentication function.

11. (Currently amended) [[An]] <u>The</u> information handling system ~~comprising:~~

~~a plurality of heterogeneous processors;~~

~~a common memory shared by the plurality of heterogeneous processors;~~

~~a first processor selected from the plurality of processors that sends an encryption request to a second processor, the second processor also being selected from the plurality of processors;~~

~~a local memory corresponding to the second processor, wherein the second processor's local memory is not shared with other processors included in the plurality of processors; and~~ of claim 32, wherein an encryption process ~~running~~ runs in the second processor, the encryption process being effective to:

[[read]] load data, associated with [[the]] an encryption request, from the common memory to the second processor's local memory;

transform the data based on the encryption request; and

write the transformed data from the second processor's local memory to the common memory.

12. (Original) The information handling system as described in claim 11 further comprising software code effective to:

read, at the second processor, one or more special nonvolatile registers, the special registers including one or more encryption keys; and

use one or more of the encryption keys in the encryption process.

13. (Currently amended) The information handling system as described in claim 11 wherein the encryption request is sent from a first processor in the at least one first processor, and wherein the sending of the encryption request ~~further~~ comprises ~~software code effective to~~:

~~write~~ writing the encryption request to a mailbox that corresponds to the second processor; and

~~read~~ reading, from the second processor, the encryption request from the second processor's mailbox.

14. (Original) The information handling system as described in claim 11 further comprising software code effective to:

identify an input data area in the common memory from which the data is read and an output buffer area to which the transformed data is written.

15. (Currently amended) The information handling system as described in claim 11 further comprising software code effective to configure the second processor by:

~~initialize~~ initializing the second processor prior to receiving the request, the initializing further including:

~~read~~ reading, from the common memory, initialization software code to be executed on the second processor; and

~~authenticate~~ authenticating the initialization software code.

16. (Original) The information handling system as described in claim 15 wherein the software code effective to authenticate the initialization software code is performed by a routine stored in a nonvolatile memory, wherein the encryption process is only performed if the initialization software code is successfully authenticated.

17. (Original) The information handling system as described in claim 16 further comprising software code effective to:

read, at the second processor, one or more special nonvolatile registers, the special nonvolatile registers including one or more encryption keys, after the initialization software code is successfully authenticated; and

restrict access to the special nonvolatile registers from outside of the second processor.

18. (Original) The information handling system as described in claim 11 further comprising:

a DMA controller associated with each of the plurality of processors, wherein the second processor reads from and writes to the common memory using DMA operations performed by the second processor's DMA controller.

19. (Currently amended) The information handling system as described in claim 11 further comprising software code effective to:

identify the encryption process and an encryption algorithm from a plurality of encryption processes and encryption algorithms based upon the encryption request; and

load encryption software code corresponding to the identified encryption process and the encryption algorithm, the load being performed by reading the encryption software code form the

common memory to the second processor's local memory.

20. (Original) The information handling system as described in claim 11 wherein the encryption process is selected from the group consisting of a decryption function, an encryption function, and an authentication function.

21. (Currently amended) ~~A computer program product stored on a computer operable media for processing encryption requests, said~~ The computer program product of claim 33, further comprising:

means for sending~~, in a computer system having a plurality of processors that share a common memory wherein at least two of the processors are dislike,~~ an encryption request from a first processor in the at least one first processor to [[a]] the second processor;

means for receiving, at the second processor, the encryption request;

means for reading data from the common memory into [[a]] the local memory ~~corresponding to~~ associated with the second processor, wherein the means for reading is performed by the second processor ~~and wherein the second processor's local memory is not shared with the first processor~~;

means for executing, at the second processor, an encryption process corresponding to the request, the encryption process being adapted to transform the data; and

means for writing the transformed data from the second processor to the common memory.

22. (Original) The computer program product as described in claim 21 further comprising:

means for reading, at the second processor, one or more special nonvolatile registers, the special registers including one or more encryption keys; and

means for using one or more of the encryption keys in the encryption process.

23. (Original) The computer program product as described in claim 21 wherein the means for sending further comprises means for writing the request to a mailbox that corresponds to the second processor and the means for receiving further comprises means for checking the second processor's mailbox from the second processor.

24. (Original) The computer program product as described in claim 21 further comprising:

means for identifying an input data area in the common memory from which the data is read and an output buffer area to which the transformed data is written.

25. (Currently amended) The computer program product as described in claim 21 ~~further comprising~~, wherein the means for configuring the second processor comprises:

means for initializing the second processor prior to receiving the request, the means for initializing further including:

means for reading, from the common memory, initialization software code to be executed on the second processor; and

means for authenticating the initialization software code.

26. (Currently amended) The computer program product as described in claim 25 wherein the means for authenticating ~~is performed by~~ operates using a routine stored in a nonvolatile memory and wherein the means for executing of the encryption process [[is]] operations only ~~performed~~ if the initialization software code is successfully authenticated.

27. (Currently amended) The computer program product as described in claim 26 further comprising:

means for reading, at the second processor, one or more special nonvolatile registers, the special nonvolatile registers including one or more encryption keys, the means for reading operating ~~performed~~ after the initialization software code is successfully authenticated; and

means for restricting access to the special nonvolatile registers from outside of the second processor.

28. (Currently amended) The computer program product as described in claim 21 wherein the means for reading and means for writing ~~steps performed~~ operate using DMA operations.

29. (Currently amended) The computer program product as described in claim 21 further comprising:

means for identifying the encryption process and an encryption algorithm from a plurality

of encryption processes and encryption algorithms based upon the encryption request; and

means for loading encryption software code corresponding to the identified encryption process and the encryption algorithm, the means for loading ~~performed~~ operating by reading the encryption software code from the common memory to the second processor's local memory.

30. (Original) The computer program product as described in claim 21 wherein the encryption process is selected from the group consisting of a decryption function, an encryption function, and an authentication function.

31. (New) A method, in a multiprocessor system, comprising:

configuring at least one first processor of the multiprocessor system to be in an shared operational state, wherein the shared operation state causes the at least one first processor to operate using a common memory accessible by a plurality of processors in the multiprocessor system;

configuring a second processor of the multiprocessor system to be in an isolated operational state, wherein the isolated operational state causes a local memory associated with the first processor to be not accessible by the at least one first processor;

executing first code within the first processor in a secure manner by virtue of the isolated operational state; and

executing second code within the at least one second processor in an unsecured manner by virtue of the shared operational state.

32. (New) An information handling system, comprising:

a common memory shared by a plurality of processors in the information handling system;

at least one first processor, in the plurality of processors, configured to be in an shared operational state, wherein the shared operation state causes the at least one first processor to operate using the common memory accessible by a plurality of processors in the information handling system; and

a second processor configured to be in an isolated operational state, wherein the isolated operational state causes a local memory associated with the first processor to be not accessible by

the at least one first processor, wherein the first processor executes first code in a secure manner by virtue of the isolated operational state, and wherein the at least one second processor executes in an unsecured manner by virtue of the shared operational state.

33. (New) A computer program product comprising a computer useable medium having a computer readable program, wherein the computer readable program, when executed on a computing device, causes the computing device to:

configure at least one first processor of the computing device to be in an shared operational state, wherein the shared operation state causes the at least one first processor to operate using a common memory accessible by a plurality of processors in the computing device;

configure a second processor of the computing device to be in an isolated operational state, wherein the isolated operational state causes a local memory associated with the first processor to be not accessible by the at least one first processor;

execute first code within the first processor in a secure manner by virtue of the isolated operational state; and

execute second code within the at least one second processor in an unsecured manner by virtue of the shared operational state.